Video Temporal Compression Techniques to Facilitate Usability Evaluation

Paulo Santos, Scott Hudson, Mark Guzdial, and Albert Badre
Graphics, Visualization, and Usability Center
Georgia Institute of Technology
College of Computing
Atlanta, GA 30332-0280

Introduction: The Problem with Video

Video of users' interaction with interfaces is a critical data source for usability analysis. Compared to other forms of usability data (e.g., think-aloud protocols, recorded log files of interface events, or interviews with users), video has the unique advantage of preserving content and context, which helps in inferring user goals and how they map to actions.

- Video shows all user actions in the interface, including both the traditional user events (e.g., button clicks, typing, dragging windows) and short but significant non-events (e.g., moving the cursor to a menu, pausing, then moving to another menu and pulling it down.)
- Video shows what the user sees, including what's typed, what error messages are generated, and how windows appear on the screen.

The problem with video is the amount of time that it requires to review and analyze. While the quality of analysis can be very high given the content and context information that is saved, the video must be studied in essentially real time, which reduces the overall efficiency of the analyst. Speeding up the video playback is not a good solution since some events would occur too quickly to be perceived by the analyst.

One solution is to index the video such that important events can be quickly and easily found and reviewed. There are two popular approaches to this task:

- **Mark the video**. Several systems (e.g., VideoNoter [Trig89, Rosc90], U-Test [Kenn89], and the Virtual VCR [Buxt90]) allow for marking the video to quickly and easily retrieve key segments. While such an approach is successful during careful review of a user's session, this approach is still time-consuming during the initial marking of the video.
- **Index based on logged user events**. If both video and user events are recorded, then the user events can be used to index the video (e.g., EVA [Mack89a, Mack89b], I-OBSERVE [Badr93, Badr94], etc.). For example, a video segment can be reviewed corresponding to when a particular error message appeared, as indicated by the events log. This approach avoids the human marking of the video, but misses information that is not recorded in or is difficult to infer from the event log, e.g., content and context information (such as what was typed when and significant non-events).

Our solution is to compress the video such that a single frame of the compressed video represents several frames of the uncompressed video. In this way, all the content and context information is preserved, while still reducing the amount of time (in terms of the number of frames) needed to perceive this information. Researchers at MIT Media Lab [Bove94] have used video compression techniques to reduce the time needed to index a video. Their techniques are used to identify large, significant events such as a scene change

or the appearance of a new character. However, their techniques are not suited to preserving the content and context information needed by the usability analyst.

In this paper, we describe several of the algorithms we've explored to compress video for usability analysis. We also present the results of a short study we undertook to determine if naive users would be able to review the compressed video and identify the interface events. Finally, we discuss the future directions of this work and other compression algorithms we plan to explore.

Temporal Compression Algorithms

The most straightforward way to compress a number of frames into one is to use simple averaging. In this scheme a pixel in the processed image is simply the average of the pixels at that position in each of the k preceding frames. This gives a temporal compression factor of k. To give a quick illustration of our notation, the algorithm for a simple average is shown below.

$$result[i] = simple\_ave[i * k]$$

$$simple\_ave[i] = \frac{\sum_{j=0..k-1} input[i-j]}{k}$$

Here *input*[*i*] represents the *i*th frame of the original input, while *result*[*i*] represents the *i*th frame of the resulting average sequence . Note that this notation expresses actions (such as addition and division) on entire frames. This is a shorthand for application of the operation on a pixel by pixel basis. Finally, it is important to note that for simplicity's sake, the boundary case of i=0 has been ignored here and in the material that follows (one would typically use *result*[0] = *input*[0] to cover this case). Another potential technique is to apply an exponentially decaying average:

$$result[i] = decay[i * k]$$

$$decay[i] = input[i] * (1 - ff) + decay[i-1] * ff$$

Here, *ff* represents a *fade factor*. This factor determines how much emphasis is placed on the current input frame and how much on the average of past frames — in other words how quickly past images fade out and are replaced by more recent images. This fading effect manifests itself in the form of *ghosting*, where old images remain behind in apparently more and more transparent form. This is most noticeable with moving objects where a fading trail is left along the path of movement. A common value for *ff* is 1/2. In that case, the current frame contributes 1/2, the next frame 1/4, the third 1/8, and so on. Higher fade factors produce faster fading, or less ghosting.

Notice that with this form of average, the rate of fading (controlled by *ff*) is independent of the compression factor *k*. This is a useful property since higher compression factors may need smaller fading factors (or higher ghosting) to allow fast actions to be perceived.

Unfortunately, both averaging techniques share an important drawback: unchanging portions of an image (such as a fixed palette, menu area, or window layout) end up being presented in their original form, while changing objects are shown in a faded fashion. This is generally precisely the wrong effect. Portions of images that do not change provide in some sense the least information to a user interface evaluator. The evaluator is typically most interested in the parts of the interface that are currently being operated upon, which in turn are the areas most likely to be changing.

This effect is illustrated in Figure 1 which shows a simple average of 10 frames within an interactive sequence (black and white images recorded digitally at 10 frames per second). Here we see that the background of the interaction is very clear — in fact much more so than the second technique to be presented below. However, the dynamic actions taken by the user are not clear. For example, we can see that a menu

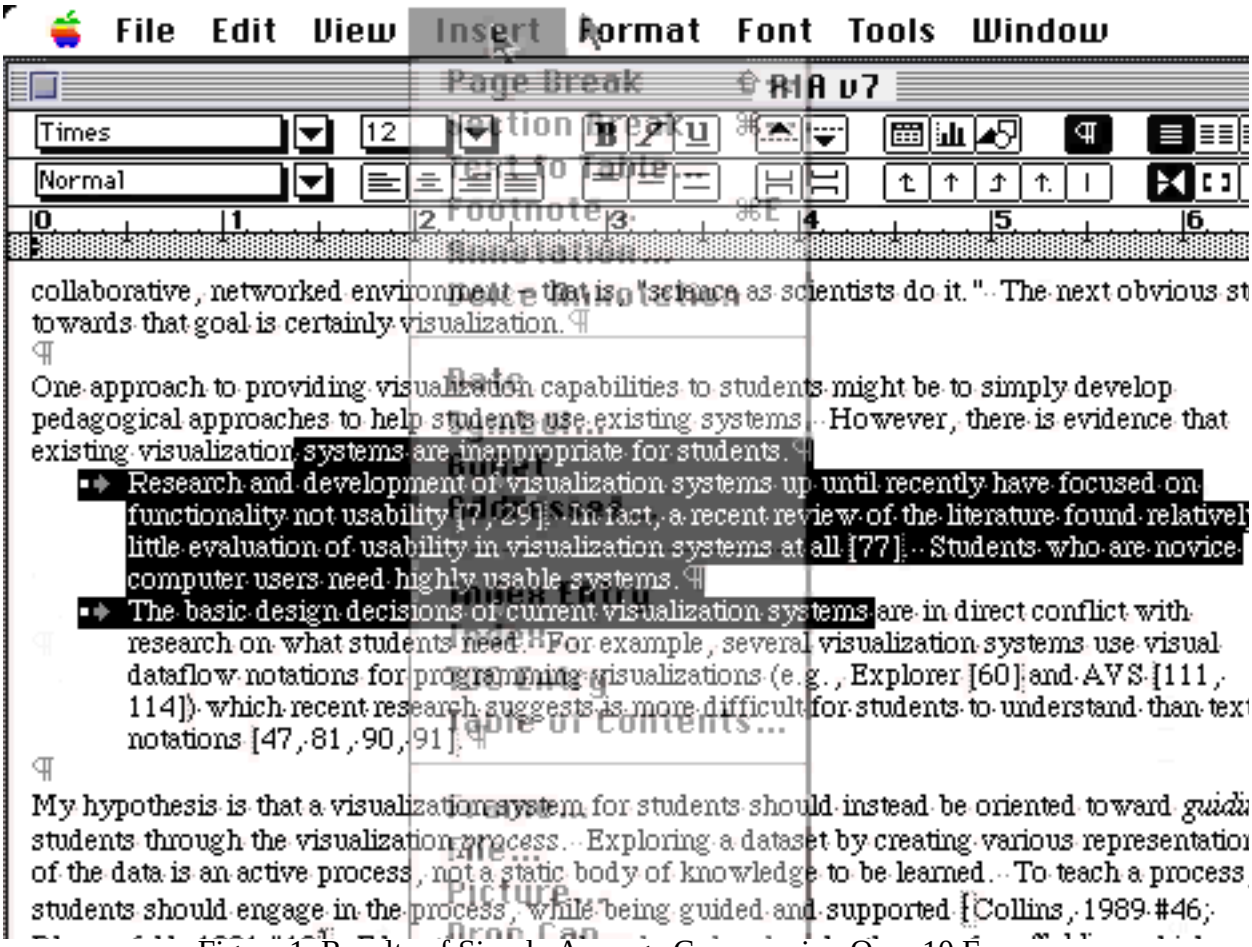

Figure 1. Results of Simple Average Compression Over 10 Frames

was pulled down, but very little is visible regarding the movement of the cursor or the selection of a specific menu item.

One would prefer a technique which presented changes most clearly and demphasized the fixed portions of an image. To accomplish this, we have constructed a new temporal compression function called *Δfade*.

This temporal compression function is designed to have properties similar to a decaying average, but with the goal of highlighting changes and demphasizing fixed components. In particular, it shows the most recently changed pixels with their actual values, while fading pixels towards a grayed image if they do not change over time. The formula for this temporal compression function is shown below.

$$result[i] = \Delta fade[i * k]$$

$$\Delta fade[i] = \begin{cases} input[i] \neq input[i-1]: & input[i] \\ input[i] = input[i-1]: & \text{fade}(\Delta\text{fade}[i-1]) \end{cases}$$

$$fade(v) = \begin{cases} v > 0.5: & (0.5 + c/2) - (v - 0.5 - c/2) * ff \\ v \leq 0.5: & (0.5 - c/2) - (0.5 - c/2 - v) * ff \end{cases}$$

If a pixel has changed† from the last frame then that pixel's value is used directly for *Δfade*. If the pixel has not changed, then a faded version of the previous image is used. The fading function used is designed to compress the dynamic range of the image to reduce its

†Since we have been working with digitally captured black and white images a simple equality test is sufficient to test for change. With smoothly changing greyscale or color images, or when using analog recording techniques that could introduce noise, a threshold test such as $|input[i]-input[i-1]| < \varepsilon$ might be more appropriate.

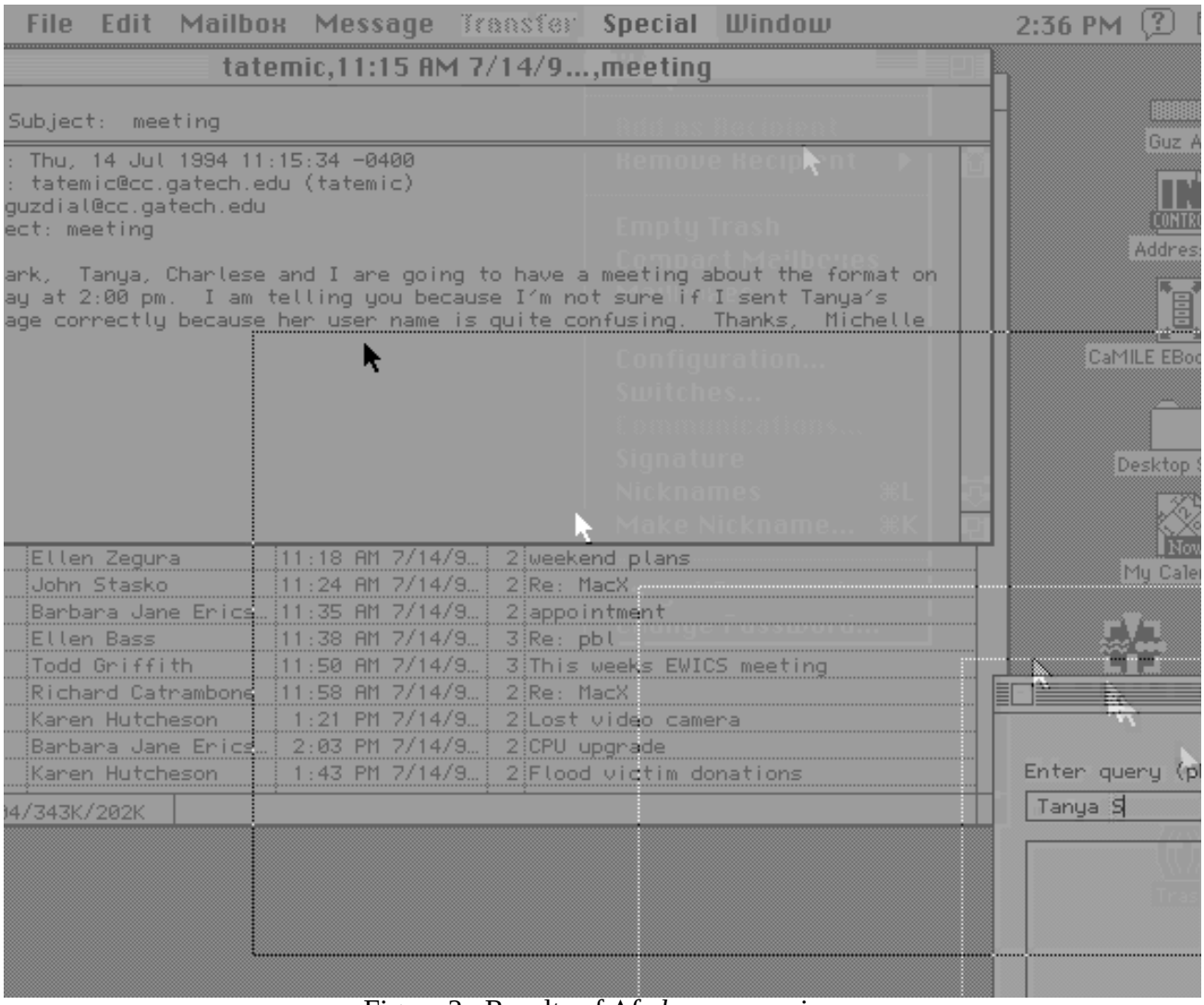

Figure 2. Results of Δ*fade* compression

contrast. Over time, unchanging images which have values in the range 0 to 1 will be faded towards equivalent reduced contrast images in the range $1/2 \pm c/2$ where c is a *maximum background contrast* parameter.

The overall effect of the *Δfade* function is to depict a recent change with its actual image, then over time reduce the contrast of the image as it remains unchanged, until it again changes and is replaced. In this way a ghosting effect that allows changes to linger on the screen can be achieved, while highlighting change rather than stability in the image.

Figure 2 shows the results of using *Δfade* compression on an actual interactive sequence. This image was produced from a series of black and white images captured digitally at a rate of 10 frames per second. A fade factor (*ff*) of 0.93 was used with a background contrast (*c*) of 0.25.

Here we can see in a single still frame that the "Special" menu was most recently used (because it is darkest), that the cursor, moved from just below that end of the menu bar to the partially visible window at the lower right hand corner of the screen, then dragged a dotted outline representing that window to-

wards the center of the screen. Note also that if we look carefully we can determine the movement direction of the cursor based on changing contrast, can see that the cursor moved quickly from the menu to the window (slightly overshooting the menu title bar), then moved more slowly near the original window position, faster in the middle of the dragging movement, and again slower as it came towards its position in the most recent frame.

By controlling the values of the parameters (fade factor, frame rate and background contrast) one can tailor the compression to the characteristics of the session and the needs of the analyst. While a high compression rate affords faster playback of the video, it also squeezes more information into a shorter time span. At some point, the rate of compression will be such that the analyst can no longer discern all the pieces of information needed for the analysis. Ideally, the analyst would have direct real-time control of the compression parameters, to allow for a selection of an optimal compression rate. Unfortunately, this is not yet feasible, as the algorithm does not run in real time.

This leads us to a discussion of the algorithm complexity. While this algorithm is of linear complexity of the number of frames, the amount of computation required for each frame is significant. For each pixel in each frame, the algorithm performs three simple computations (addition, multiplication, comparison). Since there are often hundreds of thousands of pixels per frame, a large number of operations are required. Using our sample files, it takes about 1 minute of CPU time of a Sun SparcStation 10 to process 21 frames. The good news, however, is that this algorithm is an excellent candidate for high paralellization. One can easily see that a highly -parallel system, with up to one processor per pixel, would significantly speed up the execution time.

Studying the Usability of Compressed Video

While we believe that our compressed videos successfully encode the relevant interface events, we wanted to explore if users unfamiliar with the compression algorithms were able to discern the same user events we saw in the compressed video. Our goal was not to rigorously test the hypothesis that compressed video improved usability analysis efficiency. Rather, we sought to explore if subjects could understand the compressed video as we currently were compressing the video. If subjects took more time to perceive the compressed video, missed details that were perceived in the uncompressed video, or identified events which were merely artifacts of the compression process, we would have to assume that our compressed videos were inadequate substitutes for the uncompressed video.

We digitally recorded use of two application programs on the Apple Macintosh: Eudora (an electronic mail package) and Now Up-To-Date (a calendar and scheduling system). We recorded these videos at five frames per second. We compressed 200 frames of each of these using *Algorithm #2* at a ratio of 10:1. We constructed a viewing system which supported playing the movies forward or backward at normal, half, or double speed, as well as allowing for single-stepping the movies. The viewing system also recorded log files of subjects' viewing habits and amount of time spent in the movies.

Ten subjects (volunteer students and faculty, most with usability background) participated in the study, and were each randomly assigned to one of four groups. Each group saw two movies, differing in the order of movies (Eudora or Now Up-To-Date) and compressed or uncompressed. Subjects were asked to review each movie and to write an explanation of what they perceived the user in the movie to be doing.

The data gathered from the subjects were the log files of their viewing sessions and their explanations. These were analyzed for three kinds of results:

- **Time spent studying each movie**. We wanted to know if subjects spent more time on compressed versus uncompressed videos.
- **Interface actions**. We wanted to know if subjects reviewed the compressed videos differently than uncompressed videos (e.g., used single frame more, or played them more often at half speed.)
- **Quality of explanations.** Two of the authors (Santos and Guzdial) derived goal-action trees describing what we perceived to be the hierarchy of user goals and actions in the uncompressed videos. The two other authors (Badre and Hudson), who had not previously seen any of the four videos, judged which of the goals and actions were reflected in the subjects' explanations. The judges were not told whether the subjects viewed compressed or uncompressed videos, but were told which videos were of Eudora and which were of Now Up-To-Date. The final codings were summarized in two ways. First, the branches of the goal-action trees that both judges felt that a subject's explanation touched on were recorded as an indication of the coverage of the explanation. Second, the depth of the goal-action tree that both judges felt an explanation covered was recorded. The depth was a minimum of the two judges' opinions (e.g., if judge #1 claimed that a subject perceived up to goal-action 1.3.2 and judge #2 claimed that the subject perceived up to goal-action 1.3, a depth of two was recorded.)

The results indicated that there were no notable differences between subjects review and analysis of the compressed or uncompressed videos.

- **Time:** There was no significant difference in review time between subjects viewing uncompressed (average of 359.5 seconds for the calendar application and 268.3 seconds for the electronic mail application) and the compressed videos (average of 363.5 seconds for the calendar application and 347.6 seconds for the electronic mail application).
- **Interface**: There was no significant difference in viewing strategies between compressed and uncompressed videos. Subjects viewed the movies approximately the same number of times and used play forward at normal and slow speeds the most often, in all conditions.
- **Quality of Explanations:** There was no difference in what was perceived, in either coverage or depth, between the compressed and uncompressed videos.

While our compressed videos have not yet improved the efficiency of the analysis effort, the videos are understandable by naive users. Further, the compressed videos do not lead to poorer quality (either through missed events or events generated by the compression process) or to more time for analysis. We expect that with (1) more realistically longer video segments, (2) a better understanding of the effects of the parameters (e.g., fade factor, frame rate, and background contrast) and their proper settings, and (2) better compression algorithms (see the next section), we can dramatically improve the efficiency of the analysis process through use of compressed video.

Conclusions and Future Directions

We have developed a technique for compression of human-computer interaction video sessions, with the goal of reducing playback time during usability analyses. We performed a preliminary test of the understandability of the resulting video segments, and found them to be as understandable as the full-length video. These early results will serve as a basis to build on this work, and explore its capabilities to a greater extent.

Future work on compression algorithms and systems will include: (a) extension

of the algorithm to handle color, both to deal with color information on the original frames, and to use color as part of the output coding; (b) exploration of algorithms that explore highliting of areas instead of just single pixels; and (c) improve algorithm performance, possibly through paralelization, with the goal of being able to run and compress video in real time.

Future empirical work will concentrate on validating the compression algorithms under more realistic conditions, including longer video segments (of length comparable to typical usability sessions) and specialized subjects. Through empirical work we will also study how the settings of the compression parameters affect comprehensibility and performance.

References

[Badr93] Badre, A.N., Hudson, S.E., and Santos, P.J., “An environment to support user interface evaluation using synchronized video and event trace recording”, *Georgia Institute of Technology technical report GIT-GVU-93-16*, 1993.

[Badr94] Badre, A.N., Hudson, S.E., and Santos, P.J., “Synchronizing video and event logs for usability studies”, in *Proceedings of the Workshop on Advanced Visual Interfaces (AVI’94)*, Bari, Italy, 1-4 June 1994, pp. 222–224, 1994.

[Bove94] Bove, V.M., Jr., Granger, B.D., and Watlington, J.A., “Real-time decoding and display of structured video”, in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Boston, MA, 15-19 May 1994, pp.456–462, 1994.

[Buxt90] Buxton, W. and Moran, T., "EuroPARC's Integrated Interactive Intermedia Facility (IIIF): Early Experiences", in Gibbs, S. and Verrijn-Stuart, A. A. (eds), *Multi-User Interfaces and Applications*, North-Holland, pp. 11-34, 1990.

[Kenn89] Kennedy, S. "Using Video in the BNR Usability Lab", *SIGCHI Bulletin*, **21**(2), October 1989, pp. 92-95.

[Mack89a] Mackay, W. E. and Davenport, G., "Virtual Video Editing in Interactive Multimedia Applications", *Communications of the ACM*, **32**(7), July 1989, pp. 802-810.

[Mack89b] Mackay, W. E., "EVA: An Experimental Video Annotator for Symbolic Analysis", *SIGCHI Bulletin*, **21**(2), October 1989, pp. 68-71.

[Rosc90] Roschelle, J., Pea, R., & Trigg, R., “VideoNoter: a Tool for Exploratory Video Analysis”, IRL Technical Report IRL90-0021, March 1990.

[Trig89] Trigg, R., “Computer Support for Transcribing Recorded Activity”, in *SIGCHI Bulletin* **21**(2), October 1989.